Phkmalloc

Robert C. Seacord, Software Engineering Institute [vita[1]]

2005-09-27; Updated 2008-10-06

Phkmalloc is an alternative dynamic memory management function that was by written by Poul-Henning Kamp for FreeBSD in 1995-1996 and subsequently adapted by a number of operating systems, including NetBSD, OpenBSD, and several Linux distributions.

Development Context

Dynamic memory management

Technology Context

C, UNIX, Win32

Attacks

Attacker executes arbitrary code on machine with permissions of compromised process or changes the behavior of the program.

Risk

Standard C dynamic memory management functions such as `malloc()`, `calloc()`, and `free()` [ISO/IEC 99] are prone to programmer mistakes that can lead to vulnerabilities resulting from buffer overflow in the heap, writing to already freed memory, and freeing the same memory multiple times (e.g., double-free vulnerabilities).

Description

Phkmalloc was written by Poul-Henning Kamp for FreeBSD in 1995-1996 and subsequently adapted by a number of operating systems, including NetBSD, OpenBSD, and several Linux distributions.

Phkmalloc was written to operate efficiently in a virtual memory system, which resulted in stronger checks. The stronger checks led to the discovery of memory management errors in some applications and the idea of using phkmalloc to expose and protect against malloc-API mistakes and misuse [Kamp 98]. This was possible because of phkmalloc's inherent mistrust of the programmer. Phkmalloc can determine whether a pointer passed to `free()` or `realloc()` is valid without dereferencing it. Phkmalloc cannot detect if a wrong (but valid) pointer is passed, but can detect all pointers that were not returned by `malloc()` or `realloc()`. Because phkmalloc can determine whether a pointer is allocated or free, it detects all double-free errors. For unprivileged processes, these errors are treated as warnings, meaning that the process can survive without any danger to the malloc data structures. However, enabling the "A" or "abort" option causes these warnings to be treated as errors. An error is terminal and results in a call to `abort()`. Table 1 shows some of the configurable options for phkmalloc that have security implications.

Table 1. Phkmalloc options

Flag	Description
A	"Abort." `malloc()` will terminate the process rather than tolerate failure. The core file will

1. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/274-BSI.html (Seacord, Robert C.)

	represent the time of failure rather than when the null pointer was accessed.
X	Instead of returning an error for any allocation function, display a diagnostic message on stderr and call `abort()`.
J	“Junk.” Fill some junk into the area allocated. Currently junk is bytes of 0xd0.
Z	“Zero.” Fill some junk into the area allocated (see J), except for the exact length the user asked for, which is zeroed.

After the CVS double-free vulnerability[19] , the “A” option was made automatic and mandatory for *sensitive processes* (which were somewhat arbitrarily defined as setuid, setgid, root, or wheel processes):

```
if (malloc_abort || issetugid() || getuid() == 0 || getgid() == 0)
```

A more complete description of the CVS Server vulnerability and the security implications of phkmalloc can be found in [Smashing 05].

Due to the success of pointer checks, the J(unk) and Z(ero) options were added to find even more memory management defects. The J(unk) option fills the allocated area with the value 0xd0 because when four of these bytes are turned into a pointer (0xd0d0d0d0), it references the kernel’s protected memory so that the process will abnormally terminate. The Z(ero) option also fills the memory with junk except for the exact length the user asked for, which is zeroed. FreeBSD’s version of phkmalloc can also provide a trace of all malloc, free, and realloc requests using the ktrace() facility with the “U” option.

Phkmalloc has been used to discover memory management defects in fsck, ypserv, cvs, mountd, inetd, and other programs.

Phkmalloc determines which options are set by scanning for flags in the following locations:

- the symbolic link /etc/malloc.conf
- the environment variable MALLOC_OPTIONS
- the global variable malloc_options

Flags are single letters; upper case means on, lower case means off.

References

[ISO/IEC 99]	ISO/IEC. *ISO/IEC 9899 Second edition 1999-12-01 Programming languages — C*. International Organization for Standardization, 1999.
[Kamp 98]	Kamp, Poul-Henning. “Malloc(3) revisited,” 193-198. *USENIX 1998 Annual Technical Conference: Invited Talks and Freenix Track*. New Orleans, LA, June 15-19, 1998. Berkeley, CA: USENIX Association, 1998.
[Smashing 05]	*BSD Heap Smashing*. http://thc.org/root/docs/exploit_writing/BSD-heap-smashing.txt (2005).

19. http://www.kb.cert.org/vuls/id/650937

Pearson Education, Inc. Copyright